Databasdesign för ingenjörer
Sommar 2005

En introduktionskurs i databassystem
1DL124

Kjell Orsborn
Uppsala Database Laboratory
Department of Information Technology, Uppsala University, Uppsala, Sweden
Kjell Orsborn  
Uppsala Database Laboratory  
Department of Information Technology, Uppsala University, 
Uppsala, Sweden
Active Databases

- General principles of Conventional Database Systems
Conventional (Passive) Databases

- Data model
- Transaction model
  - ACID principle
- Examples of real world problems:
  - Inventory control
    - reordering of items when quantity in stock falls below threshold.
  - Travel waiting list
    - book ticket as soon as right kind is available
  - Stock market
    - buy/sell stocks when price below/above threshold
  - Maintenance of master tables (view materialization)
    - maintain table that contain sum of salaries for each department
Conventional Databases

- Passive DBMS
- Periodical polling of database by application
  - Frequent polling => expensive
  - Infrequent polling => might miss the right time to react
  - DBMS does not know that application is polling.
Active Databases

- Active DBMS
- Recognize predefined situations in database, application, environment
- Trigger predefined actions when situations occur, such as DB operations, program executions
Active Databases

- General idea
- ADBMS provides:
  - Regular DBMS primitives
  - Definition of application-defined situations
  - Triggering of application-defined reactions
Active Databases

- Possible applications of ADBMS:
  - Consistency enforcement
    - Reaction to violations
      - e.g. ROLLBACK when constraint violated
    - Connection to time
      - e.g. check some constraint violations every midnight
  - Computation of derived data
    - View materialization of derived data
      - e.g. incremental recomputation of view of sum of salaries per department, salsum(dno,salary), computed from employee(ssn,dno,salary)
      - ... or invalidation of materialized view when relevant update (e.g. ssalary) occurs => rematerialize view when accessed next time
  - Automatic travel booking
    - Order currently not available ticket with desired properties whenever one gets available
Active Databases

- Semantics of ECA rules
  - Most common model presently
  - Event Condition Action:
    - WHEN event occurs
    - IF condition holds
    - DO execute action
Active Databases

- Example - Oracle syntax
  - `EMPLOYEE(SSN, DNO, SALARY)`
  - `DEPT(DNO, MGRSSN)`
  - `SALSUM(DNO, TOTAL) <= Materialized`
  - `CREATE TRIGGER EMPLOYEE_SALARY_MATERIALIZATION`<--- Event
    `AFTER UPDATE ON EMPLOYEE`<--- No condition
    `FOR EACH ROW`<--- Action
    `BEGIN`<---
    `UPDATE SALSUM S`<---
    `SET TOTAL = TOTAL - OLD.SALARY`<---
    `WHERE S.DNO = OLD.DNO`<---
    `UPDATE SALSUM S`<---
    `SET TOTAL = TOTAL + NEW.SALARY`<---
    `WHERE S.DNO = NEW.DNO`<---
    `END;`<---
Active Databases (ECA)

- **Event:**
  - Update of database record(s)
  - Parameterised using pseudo tables OLD and NEW

- **Condition:**
  - Query on database state,
  - e.g. a database query
    - empty result => condition is FALSE
    - non-empty result => condition is TRUE

- **Action:**
  - Database update statement(s)
  - Stored procedure execution

- **Unconditioned (EA) rules, as in example:**
  - ON ... DO
  - Natural in C++/Java based object stores
  - Can make arbitrary C++/Java test in DO part!

- **Condition/Action rules:**
  - Not common in databases
Active Databases

- Example of triggers for constraints, Oracle
- CREATE TRIGGER SALARY_CONSTRAINT
  AFTER UPDATE OF SALARY ON EMPLOYEE <--- Event
  WHEN NEW.SALARY > <--- Condition
  (SELECT M.SALARY
   FROM EMPLOYEE M,DEPARTMENT D
   WHERE NEW.DNO = D.DNO AND D.MGR = M.SSN)
  BEGIN <--- Action
  UPDATE EMPLOYEE E
   SET SALARY = OLD.SALARY
  END;
Active Databases

- NOTICE! SALARY_CONSTRAINT needed for managers:
- CREATE TRIGGER SALARY_CONSTRAINT2
  
  ```
  AFTER UPDATE ON EMPLOYEE
  WHEN NEW.SALARY <
    (SELECT E.SALARY
     FROM EMPLOYEE E, DEPARTMENT D
     WHERE E.DNO = D.DNO AND D.MGR = M.SSN)
  BEGIN
    ROLLBACK
  END;
  ```

- NOTICE! SALARY_CONSTRAINT needed for departments too!
- NOTICE! SALARY_CONSTRAINT needed for insertions too!
- Solution: Integrity constraints!
Active Databases

- Advanced level SQL-92 (SYBASE, ORACLE) has assertions too:
- CREATE ASSERTION SALARY_CONSTRAINT
  CHECK(NOT EXISTS
    (SELECT *
     FROM EMPLOYEE E, EMPLOYEE M,DEPARTMENT D
     WHERE E.SALARY > M.SALARY AND
     E.DNO = D.DNO AND
     D.MGRD = M.SSN))
Active Databases

- Cautions:
- Very powerful mechanism:
  - Small statement => massive behaviour changes.
  - Rope for programmer.
  - Requires careful design
  - *Activity design* + traditional database design.
- Trace consequences of rule specification/changes:
  - Make sure indefinite triggering cannot happen.
- Help user design meaningful rules:
  - Tool, e.g. simulator
- Higher abstraction level needed for ordinary users:
  - Assertions.
  - Specification of materialized views.
  - Graphical tools.
Active Databases

- Workflow Control Application
- Goal: Control long running activity
  - multiple application steps
  - may be of long duration
  - may be executed by different servers in a distributed system

- Issues:
  - how to sequence the steps (workflow)?
  - if a step fails, committed earlier steps cannot be rolled back: how to compensate?

- Approach: use triggers/rules
  - model application steps by transactions
  - use rules to check constraints, chain steps, and handle exceptions
  - flexible response to exceptions, rather than a fixed compensation policy
Active Databases

- SUMMARY
- Active DBMSs embed situation-action rules in database
- Support many functionalities:
  - E.g. Integrity control, access control, monitoring, derived data, change notification
- Some ADBMS functionality commercially available as triggers: Sybase, Oracle, Interbase, etc.